Design of the DECmcc Management Director

1 Abstract

The DECmcc product family represents a significant achievement in the development of enterprise management capabilities. DECmcc embodies the director portion of Digital's Enterprise Management Architecture (EMA) and is both a platform for the development of new management capabilities and a vehicle for aiding customers to manage their computing and communications environments. Initially, the DECmcc director was intended to facilitate sophisticated management of evolving networks. In addition to network management, DECmcc has been adapted to the needs of system, applications, data, environment, and telecommunications management. The first implementations contained the DECmcc kernel, a developer's toolkit, and various management modules.

Development of the DECmcc director has been a multiyear effort involving many groups within Digital. When the DECmcc design was initiated in 1987, there was no equivalent management software in the industry. Most companies, Digital included, provided one or more independent, focused products. Each of these dealt with managing a specific set of components such as a single vendor's local area network (LAN) bridges or providing a specific management application such as equipment inventory.

Digital's network management capabilities within DECnet Phase IV were reaching their limit, and the incorporation of newer communications technologies in a seamless way was becoming increasingly difficult. As part of the DECnet Phase V development, work was started to rationalize management of distributed systems. This effort led to the formal definition of such concepts as the director/entity relationship, the Entity Model, and the common management information protocol (CMIP).[1,2,3,4] These ideas formed the basis for management in Phase V and were Digital's contributions to the open systems interconnection (OSI) management model from the International Organization for Standardization (ISO).

The original vision of network management in Phase V included the concept of two management directors. The first, a sophisticated director referred to as the management control center (MCC), would handle the more complex, yet user-oriented, management tasks. The second, a simple command line director referred to as network control language, would address the needs of more experienced managers who prefer a command line environment.[5]

Conceived primarily as a DECnet management director, the DECmcc product evolved to address the broader problems associated with managing a complete computing and communications environment. This evolution is not yet finished and arguably will never finish as network environments continue to

change.

Design of the DECmcc Management Director

Since the development of DECmcc in 1987, the simple network management protocol (SNMP) has become widely implemented. DECmcc has adapted to handle SNMP as well. In addition, the DECmcc product, once a tool for the VAX VMS architecture, is now implemented on multiple platforms, such as the ULTRIX and UNIX System V Release 4 operating systems.

In this paper, we look at the development of the DECmcc director. We start by discussing our initial design ideas taken in the the perspective of the industry at the time. We then describe the initial implementation of DECmcc. We also present the effects of the changing industry and how DECmcc has adapted over time. We conclude with some of the opportunities for future work.

2 Historical Perspective

Digital's first network management capability was delivered in 1978 as part of the release of DECnet Phase II software. Much of the DECnet product was then manageable, both configuring the software for installation as well as the operational aspects. The main program used to perform management was the network control program (NCP). At that time management mostly consisted of looking at information and then changing it as needed. DECnet Phase II, however, could perform sophisticated diagnostic loopback tests, both nonintrusive as well as intrusive, to diagnose connectivity problems at various layers of the protocol stack.

Management formed a significant part of the DECnet Phase III and DECnet Phase IV networking products. Each major release contained many changes to manage the new functionality. However, the DECnet management structure in place in the 1970s was becoming more difficult to adapt to the requirements of the mid-1980s. For example, support was added for X.25 during Phase III and for Ethernet during Phase IV. These releases required quite different management approaches than the one used for Phase II. With the advent of the significant changes to DECnet Phase V to include support for the OSI protocol stack, another management approach was needed.

Thus in conjunction with Phase V network development, an effort was started to provide a new architectural approach to management of Phase V. One of the key requirements was to provide the Phase V management needs in a way that would extend their adaptability to the future. This work was referred to as distributed systems management because it addresses management of the computing environment as well as management of the communications that DECnet comprises. Most of the initial work in distributed systems management concerned itself with the aspects that applied to DECnet and the changes needed to provide manageability of DECnet in Phase V. The primary underlying concepts were articulated.

- Directors are management programs used by human managers to effect

management. Entities represent managed components to directors through software referred to as agents.[6]

- The Entity Model is the underlying model for managed entities defined in terms of an object-based approach.[1,3,7]

- The formal specification for the classes of entities is defined in terms of Module-2+ like specifications and is called management definition language (MD).[3]

- A command language, network control language (NCL), was formally defined to be unambiguous even with new entities and their definitions; an associated primitive director of the same name, part of every Phase V package, replaces the NCP of previous phases.[5]

- A management protocol called the common management information protocol (CMIP) was used to communicate between directors and entities.[4,8,9]

  CMIP was named common and presumed to handle the common aspects of management across a wide variety of management applications. Some developers suggested the possible need for a small number of specialized management information protocols (SMIPs) - perhaps one for each of the management functional areas (configuration, performance, fault, security, and accounting). However, CMIP proved to be sufficiently expressive and powerful to support management applications covering the management functional areas.

At the time the distributed systems management work was initiated, Digital's networking and communications product line was expanding to encompass more than the DECnet networking hardware and software. Along with each product came its own management software, some of which was tailored along the lines of the DECnet standard NCP. In addition, the Network Management Development Group was building some fairly sophisticated management applications that went far beyond the capabilities of NCP in DECnet. The developers necessarily took a different approach to management.

Thus, by the late 1980s Digital had developed a number of distinct management products. Many of these employed private protocols, for example

- NCP for managing DECnet, based on a command line user interface

- NMCC/DECnet monitor, a wide-area DECnet monitoring tool, based on a graphical user interface

- NMCC/ETHERnim, an Ethernet monitoring/inventory test program, based on a graphical user interface

- RBMS, Remote Bridge Monitoring Software for managing Digital's bridge family, based on a command line user interface similar to NCP

- TSM, Terminal Server Manager for managing Digital's terminal server family, based on a command line user interface similar to that used in the terminal servers

- LTM, LAN traffic monitor for understanding the traffic usage and patterns of Ethernet segments, based on a graphical user interface

Other manufacturers also provided management software capable of managing their devices. Some vendors provided particular management applications that were not tied to any specific network device. These applications performed a single function, such as maintaining an inventory of equipment on behalf of a manager.

The plethora of management capabilities from many vendors created many choices for end users. At the same time, the diverse applications were perceived as carrying significant drawbacks. Each application provided its own user interface. Each had its own database for storing management information. Each dealt with different management information. In addition, each tool provided its own, often rudimentary, independent management application.

End users viewed these many products as creating a series of problems: (1) A manager needed multiple management terminals, one per product. (2) Separate training was required to use each product. (3) Confusion occurred when the user switched between multiple products. (4) Different information was available from each product, or worse, the same information was available in a different form. (5) There was no ability to share information between products. (6) It became difficult to diagnose problems that spanned multiple technologies. Other aspects of the system management perspective in 1986 have been described.[10]

At that time, standards for network management had not progressed very far; SNMP did not yet exist. In fact, agreement on the overall concepts had only begun within the OSI management committees.

It is with this background, then, that the design of DECmcc as a management director was undertaken.

3 Opportunities

Of all the situations that existed in customer networks in the mid-1980s, probably the most important was the realization that networks no longer consisted of equipment from a single vendor. In addition, different technologies were commonly used to improve a given customer's network. With each technology came its own management protocol, along with its own management structure. As networks became larger, more than one network manager was typically needed.

The opportunity existed to provide complete, integrated network management that could be adapted to the changing needs of management. Our product goals were

- To provide a consistent, integrated user interface, permitting management of any component in the enterprise to be performed in a style

that does not depend on the specific component

- To provide integration of the management data (contained in the components as seen by the director) and management information (as constructed by the director using the management data)

- To provide a consistent, extensible means of storing management information and of allowing it to be accessed conveniently by multiple independent management applications

- To provide an application programming interface (API) to support management applications

Obviously, an approach necessary to solve these nontrivial problems was not to be a small undertaking; an architected approach was appropriate.[6]

4 Design Approach

The solution to the problems outlined was seen to be a distributed applications environment, tailored to the specific needs of management. Quite quickly, the idea of defining a modular and extensible environment was selected.

Management capabilities could be added in a straightforward fashion based on an applications kernel, which could either be replicated as needed around a network, or considered as multiple, cooperating kernels supporting a distributed management environment. Hence a kernel with modules that can be added dynamically, much as applications are added to an operating system, is fundamental to the design of DECmcc.

The next consideration concerned the composition of the modules themselves. One approach to the support of multiple technologies had one module access each different sort of component to be managed. Since a number of management application functions were desirable, one might have a module for each such function. Also one might have a module for each form of user interface to accommodate the different user interface styles, such as command line or windowing.

Thus, we arrived at the concept of distinguishing form, function, and access. Furthermore, we defined management modules based on presentation modules (PMs) for user interface, function modules (FMs) for management functions, and access modules (AMs) for accessing each distinct technology. The DECmcc director structure is shown in Figure 1.

We observed that the EMA Entity Model, defined initially to meet the needs of management of entities, provided generalized structuring concepts that would be appropriate for the director environment as well. Indeed, choosing the same model to handle the needs of the director removed the need for a translation between the entity environment and the director environment for EMA entities, which has proved to be advantageous for the implementations. Hence the following Entity Model concepts were also used in the director.

- An object-oriented approach-encapsulating objects (entities) and their

operations

- A class structure-defining attributes, operations, and events for each class and specifying management information using a management specification language

As we studied the needs for stored management information in the director, we identified four different sorts of information, distinguished by the storage needs, nature of the contents, and the access patterns.

1. Class data-the dictionary of all management operations, attributes, notifications, and their related definitions categorized by class, updated infrequently, but read often

2. Instance data-the configuration information, stored in a global naming service, changing often, but read from many places simultaneously

3. Historical data-information about specific entity instances stored over time, written incrementally and read sporadically according to the needs of applications using such data

4. Miscellaneous data-other data needed for specific modules, such as tariff information or the definition of rules specifying alarm conditions

The complete logical information store was termed the management information repository (MIR).

The kernel defines an execution environment that is suitable for management modules and supports the MIR. This was initially implemented in terms of technology provided completely within the director kernel. Many of the kernel services, however, were subsequently replaced with distributed systems services, including multithread support, naming/directory service, time service, and remote procedure call (RPC).

It is, perhaps, interesting to note that the decision to use a multithreaded approach in DECmcc was not unanimous. The alternate approach proposed an asynchronous message-passing scheme. Although the decision to use a multithreaded environment has proved to be implementable, we did not appreciate how the performance of the multithreading implementations would affect the ability to support the needs of application environments such as DECmcc.

Invoking Module Services

As we looked at how management modules would call each other, we chose a fairly straightforward approach. User interactions with a PM would cause the PM to invoke an FM, the FM to then invoke the appropriate AM, and the AM to communicate with the desired entity. The response would then be transmitted through the AM, FM, and PM, with the result presented to the user. Thus the simple procedure call paradigm between modules, as shown in Figure 2, supported the needs of applications geared toward monitoring and control operations.

However, one must consider the increase in the total number of management modules over time, and the even greater increase in the total number of available management services (defined by specific operations on classes of entities). Thus, it became clear that the intermodule procedure calls could not use named procedures, as administering the names of ever

increasing numbers of procedures would be a burden. Instead we chose an approach whereby modules invoked each other's services by referring to the operations and the objects, using a service invocation procedure known as "mcc_call." We defined the interfaces provided by the management modules entirely in terms of operations on objects - an object-oriented approach - but this approach did not require the use of object-oriented languages or databases.

We further observed that one could decompose a management application into a number of smaller, potentially reusable services. Hence FMs could invoke other FMs in performing their services much in the same way that applications on UNIX systems pipe results from one component to another. Given the generally extensible nature of DECmcc and the supporting mcc_call structure, this led to the concept of generic applications. Being run-time driven from the class dictionary, these applications could work over a wide range of managed objects and perform the same service for each of them without a priori knowledge of the objects. For example, one might have an FM that provides performance-related services, turning error counters (obtained directly from the managed objects) into error rates (by simply polling for two counter values, subtracting one from the other, and dividing by the time interval between polls). A different FM might provide alarm services by notifying users of particular (user-specifiable) conditions, such as when a particular counter exceeds a defined threshold.

Of course, managers are often more interested in error rates exceeding a given threshold. The same alarms FM could be primed to look for an error rate; the request would be passed on to the performance FM, which in turn would calculate the rate by looking at successive polls of the error counter. The alarms FM does not need to be aware whether the data it needs comes from the performance FM or directly from the managed object via the appropriate AM. The disposition of the methods among modules is hidden by the service invocation mechanism.

Furthermore, the alarms FM tracks the number of times a user is notified of a problem, and this counter is available as management data. One might then want to determine the rate of user notifications (using exactly the same generic performance FM as before), and use the same alarms FM to notify a different user when the rate of notifications exceeds a defined threshold. This threshold might indicate that one manager is being overloaded. Thus, in this scenario we have a number of modules involved in a calling hierarchy, with the same modules appearing more than once. Figure 3 shows the reuse of software using generic function modules in DECmcc.

Management Specification Language

The entity model's management definition language, originally intended for the specification of management agents, was modified and applied to

the director environment. Director-oriented information was added to the management specification, such as user interface tags for automatically generated forms and menus. This information was named the management specification language (MSL). An MSL compiler was defined to convert MSL to

an on-line form, available as metadata through an on-line dictionary, the MIR class data. With the management specification information available to management modules, modules could adapt their behavior as new modules were added; this is especially important for generic modules. Thus the same MSL that was used to help the entity agent developers was also useful for the management director to drive the extensible management modules.[11]

This dictionary information spurred the definition and development of the generic management modules. The generic PMs provide an extensible user interface that is capable of adapting as new managed objects or applications are added. The generic FMs provide consistent functions over a broad set of managed objects. Finally, the generic AMs support extensible management protocols, allowing the dynamic addition of new sorts of managed objects.

The design of the DECmcc director led to a number of possibilities in the type and application of the different sorts of modules. Initially AMs were conceived as being one per management protocol, which usually translated to one AM per type of device (such as bridge, terminal server, DECnet node). Since the advent of standard protocols, such as SNMP from the Internet community and CMIP for OSI management, AMs are now more typically generic and extensible.[8,9,12] A single AM covers many different types of device with one protocol.[13]

For FMs, we originally envisioned two sorts of modules: the generic FM providing the same function over a wide variety of managed objects, and a specific FM providing a set of functions for a single class of managed object. Today, we believe one may have two different sorts of generic FM: one that is specific to a technology (such as network management related), and another, truly generic, which is completely independent of the technology being managed (such as an alarms FM).

For PMs, we recognized the need to handle device-specific aspects as well as user interface style-specific aspects. Normally one would have generic PMs provide user interface capabilities over a broad variety of managed objects and applications. However, to support the specific needs of generic FMs, specific PMs might be used to provide the appropriate user interface. PMs that are specific to an FM are less useful since they do not provide a consistent user interface "look and feel."

During the design of the DECmcc director, a number of smaller, but nonetheless important, design decisions were made. The concept of management domains was defined as a general container mechanism for entities, which could include domains themselves. Domains therefore provide a flexible, user-specifiable organizational structure for both visual representation at the user interface, as well as a means to organize the stored management information and associated background processing.[14] The

need to provide a consistent approach to the naming of objects within the director was established. This was initially based on Digital's distributed name service, DECdns, providing globally unique names and network-wide access to those names.[15] Finally, the concept of time, including the

scheduling of operations as well as scope of interest for information retrieval, was included in the mcc_call API. The time concept allows management applications to be developed that can operate on historically stored information as easily as they can on data retrieved directly from the network.[16]

A more detailed report on the design of DECmcc has been published.[17]

Some other aspects of the DECmcc program, while not part of the technical design, had a major part to play in its evolution. First was the need to provide published, open definitions of the DECmcc API, based on existing standards. This allows other vendors and end users to develop their own management capabilities to add to DECmcc. Second was the establishment of a strategic vendor program within Digital to work with other vendors, particularly those that provided network technologies that complemented Digital's own offerings, to help them develop to the DECmcc platform. Finally a design center program was instituted whereby the design of DECmcc would be validated, as it evolved, against the needs of some major customers to ensure that it continued to address the management problems of those customers.

5 Broadening the Scope

Since DECmcc was designed to be able to manage anything that could be described by the entity model, and since the entity model is a general object-oriented framework, it follows that it is feasible to extend DECmcc to classes of managed object and applications beyond the traditional network-oriented view of nodes, hosts, bridges, routers, etc. Some of the new classes of managed objects and new applications that we have seen developed using DECmcc include

1. Management of applications such as transaction processors and databases

2. Applications in traditional system management, such as user management, disk backup, software installation, configuration maintenance, and performance monitoring

3. Management of objects in the telecommunications field, such as PBX machines, multiplexers, and switches[18]

4. Management of noncomputer hardware, such as air conditioners and building-environment controls

Note that the implementation of these extensions generally involves a relatively small investment, at which point the power of existing generic applications is automatically provided. For example, in the easiest case, a new object that is manageable through SNMP need only have its management

information base (MIB) translated to MSL and loaded into the DECmcc dictionary, at which point it is accessible by the existing SNMP AM as well as the standard generic applications.

In other cases, such as the air conditioning example, it is only necessary to code an AM that communicates to the air conditioning controller through its private protocol. Functions such as alarms, notifications, historical data recording, and graphing are automatically provided by existing FMs and PMs upon recognition of the new object class.

In complex cases, object-specific FMs are written to perform such tasks as software installation and disk backup control. Yet even in these cases, all these functions are automatically accessible through the generic PMs.

The potential for interdisciplinary applications is now becoming possible by the normalization of the interfaces to objects traditionally handled by totally separate applications. For example, given the extensions described above, it is possible to write an application that activates an emergency disk backup and switches telephone trunk traffic to another building if an air conditioning failure occurs. In fact, depending on how the various objects are defined, it may even be possible to create such an application simply by writing a single alarm rule.

Evolution to Open Systems

With recent industry trends toward open systems environments, as well as the realization that almost any enterprise now comprises multiple hardware and software platforms from multiple vendors, it was clear that DECmcc had to evolve to this new world. Among the requirements to be met were not only the management of objects existing on various platforms, but also the execution of the director itself on different hardware and operating system platforms.

These requirements dictated two basic design goals:

1. Portability of the director kernel itself to environments other than VAX VMS

2. Portability of plug-in management modules to a DECmcc director running on any supported platform, and in particular, source compatibility to the greatest extent possible with the considerable suite of management modules that existed when the porting effort started

Many of the fundamental requirements for portability had already been met. All existing management modules were coded to the API defined in the {ital}DECmcc System Reference Manual{xital} (SRM), and the SRM had little code that was inherently specific to VAX or VMS.[19] In fact, only the documented SRM routines were used to access DECmcc services, as well as many other common operating system services such as data storage and thread control. Consequently, the kernel implementation team had the flexibility to implement these services differently on various platforms without impacting management module source code. This was particularly true with the all-important mcc_call service, which provided the API for intermodule communication in a platform-independent context such that a wide variety of interprocess or intraprocess communications mechanisms could be chosen for the underlying implementation.

In the initial porting effort, which was from VAX VMS to RISC (reduced instruction set computer) and VAX ULTRIX, some of the more important changes in underlying implementations were

1. The MIR was implemented over the ndbm hash database manager. An earlier version of the MIR was also implemented over ULTRIX SQL, which provided some large-capacity database features at the expense of significant performance.

2. The operating system time interfaces were migrated to the distributed time service of the Open Software Foundation distributed computing environment (OSF DCE).

3. The multithreading services were migrated to the DECthreads component of the DCE.

4. The intermodule communication mechanisms (mcc_call) were implemented using RPC technology, with management modules running as independent RPC server processes. This allowed run-time extensibility without requiring the operating system to support a merged image activation function, a feature of the VMS implementation.

5. Through the use of various wrapper routines in the DECmcc development toolkit, we were able to allow the management module developer to code entry points to the management modules without distinction to whether they were being run in an image merge or an independent process context.

Despite these major changes, 85 percent of the kernel code is in fact platform independent, and we are maintaining a single source pool for DECmcc regardless of the number of platforms. To minimize the operating-system-dependent code we must maintain and to provide backward compatibility, we are also porting to VMS a number of the above technologies such as those built on DCE.

At the present time we continue to broaden our open systems focus by additional ports to UNIX System V, OpenVMS on Alpha AXP, OSF/1 on Alpha AXP, as well as other operating systems.

6 Implementation

In late 1990 and early 1991, Digital delivered the first two versions of DECmcc. Version 1.0 was written to allow other vendors to start building their management modules; version 1.1 added some components for network managers. Both releases ran on VAX VMS systems, either workstations or hosts.

In the middle of 1992, Digital released version 1.2 of DECmcc, which added significant capabilities and runs on RISC ULTRIX. Later in 1992, Digital delivered POLYCENTER SNA Manager. In conjunction with DECmcc and the SOLVE:Connect for EMA, a product from System Center, Inc., it allows bidirectional management between IBM SNA hosts and DECmcc systems.[20]

In early 1993, Digital released version 1.3 of DECmcc under the new product family name of POLYCENTER, with the POLYCENTER Framework, which is the basis for POLYCENTER Network Manager 200 and POLYCENTER Network Manager 400. This new version adds ways to provide simpler, yet powerful, integration of management capabilities; uses an OSF/Motif graphical user interface; and provides additional development tools. These versions contain the DECmcc kernel, a corresponding developer's toolkit, and a series of management modules, which are outlined in Table 1. The SRM provided the API definitions for management modules, as provided by the kernel. Figure 4 shows a sample screen from DECmcc being used to manage a portion of a network.

NOTE

Figure 4 (Screen Display of DECmcc version 1.3) is a photograph and is unavailable.

Since the DECmcc kernel is indifferent to the specific type of any management module, it is quite convenient to package different modules together, providing for a flexible packaging scheme. Each DECmcc can therefore be tailored to include the set of modules appropriate for managing the environment in which it is situated. In addition, modules from other vendors can be integrated by the customer without involvement from Digital.

As new management modules are added, the powerful generic capabilities of DECmcc allow many existing functions to be used without change. When an AM is added for a new class of resource, or when an existing generic AM is enhanced by adding new supporting definitions in the dictionary, one can immediately perform the following functions.

- Identify specific resource instances uniquely
- Make the resources known to all DECmcc directors in the network
- Represent the resources on an iconic display in one or more management domains
- Examine management attributes from these resources
- Modify management attributes in these resources
- Apply management actions to these resources
- Display event information from these resources
- Create alarm rules that can be triggered on particular conditions (polled or unsolicited) about these resources
- Have the relevant icons change color when the alarms fire
- Store, periodically, management data or information about these resources in the DECmcc historical data store, or export the information to a relational database
- View the stored historical data
- Process the relational data using standard information management tools, for example, to provide management reports

Of course, work on a major software system such as the DECmcc director is never complete. There are many areas of opportunity for additional development. For example, DECmcc can be ported to other industry platforms (both hardware and software). New objects can be managed, not only in

network management but also in system management, application management, data management, environment management, telecommunications management, and so on. Commensurate with each of these general areas are technology-specific applications. In addition, further technology-independent generic applications can be developed. A recent paper describes how DECmcc can be considered as a distributed application and some additional work to make use of the DECmcc concepts in a distributed environment.[21]

DECmcc is not the only management director in the industry. Thus interoperability between DECmcc and other management systems is another area of opportunity. DECmcc already has links to other management systems, not the least being to manage IBM SNA systems.

Recent advances in object-oriented technology can be incorporated to enhance the object orientation of DECmcc.

Finally, new standard industry management protocols, new managed objects, and management framework innovations are always becoming available. DECmcc will be taking all of these evolutions in its stride. The distributed management environment (DME), still under development by OSF, promises to bring yet more technology to which DECmcc will adapt readily.

Table_1:_DECmcc_Director_Management_Modules

Presentation Modules	Definitions
Forms and Command Line PM	Provides a command line user interface based on the NCL definition, together with a full-screen mode for video terminal devices. This PM also executes DECmcc command scripts.
Iconic Map PM	Provides an iconographic display based on OSF/Motif. It supports all the capabilities of the command line, but with a more usable graphical representation of the network and pull-down menu support. This PM also provides on-line graphing of management information. In addition, this PM can launch management applications that are not strictly part of the DECmcc environment, to provide a visual integration for the manager.
Notification PM	Provides an interactive management display of event or alarm firing conditions based on OSF/Motif. Flexible filtering of information is used to minimize the information displayed to the manager, but the manager can search for and display information using various criteria such as severity level, managed object, and data and time.
Function Modules	**Definitions**
Registration FM	Provides a means for registering entities with the director and for maintaining reference information on behalf of the entities.
Domain FM	Maintains the definitions of the various management domains, their membership, and their relationships.
Historian FM	Enables the capture and storage of user-specified management attributes from any entity in the network. Retrieval of the stored information by management modules is provided directly by the mcc_call API.
Exporter FM	Allows extraction of user-specified on-line or stored

management information into a relational database for processing by SQL-based information management tools, such as reports.

Table 1 (Cont.): DECmcc Director Management Modules

Alarms FM	Permits managers to specify, through rules, the set of conditions about the network in which they are interested. When the alarms FM detects a condition (the rule fires), various notification techniques may be employed. These include invoking a command script, sending mail, calling a manager using an electronic beeper, or modifying an icon on the iconic map display.
Performance Analyzer FM	Calculates statistics for DECnet, transmission control protocol/internet protocol (TCP/IP), and LAN bridges, based on error and traffic utilization or other information.
Diagnostic Assistant FM	Helps the manager diagnose faults in a TCP/IP network, based on some of the more frequently occurring TCP/IP network problems.
Autoconfiguration FMs	Determine automatically the configuration and topology of specific portions of the network. Included are FMs to determine the configuration and topology of DECnet Phase IV networks, IP subnetworks, fiber distributed data interface (FDDI) ring maps, and LAN bridge spanning trees.

Access Modules	Definitions
SNMP AM	Provides access to objects that implement the SNMP protocol. It is a generic AM in the sense that it can adapt to new object definitions using information in the DECmcc dictionary. New MIB definitions are provided in a standard form and translated by a MIB translation utility into the DECmcc dictionary.
DECnet Phase IV AM	Provides access to the DECnet Phase IV implementations, be they hosts or servers such as routers. This AM implements the network information and control exchange (NICE) protocol.
DECnet/OSI Phase V AM	Provides access to the DECnet/OSI Phase V implementations, hosts, and servers. It implements the CMIP protocol used in Phase V.

Bridge AM	Supports Digital's family of LAN bridges, including the LANbridge 100, LANbridge 150 and LANbridge 200, and the DECbridge family. It implements the RBMS protocol, which is used by the original management product of the same name.
Access Modules	Definitions

Table 1 (Cont.): DECmcc Director Management Modules

FDDI AM	Supports Digital's FDDI DECconcentrator products and other devices that support the standard station management protocol (SMT).
Terminal Server AM	Supports Digital's family of terminal servers, implementing management through the maintenance operations protocol (MOP).
Ethernet Station AM	Supports all Ethernet and IEEE 802.3 stations that implement either, or both, the Digital MOP protocol or the IEEE 802.2 XID and TEST messages.
Circuit AM	Uses the services of other AMs to provide management of the network circuits that connect systems together, based on DECnet nodes, TCP/IP hosts, or network management forum definitions. Such circuits might be simple point-to-point or could represent complex multichannel circuits.
SNA AM and Agent PM	Permit bidirectional management of the SNA environment and the DECmcc management environment through a component that resides on an SNA host (either IBM's NetView or System Center's Advanced System Management).
Data Collector AM	Provides a means to allow other software, such as applications, to send events into DECmcc so they may be processed and analyzed along with events from devices or applications that have access modules.
Script AM	Allows invocation of existing or custom shell scripts or command procedures from DECmcc, and information to be returned from the scripts into DECmcc for processing and analysis by other modules.

7 Summary

This paper has explained aspects of the design of DECmcc in the context of the state of the industry at the time. DECmcc has been a large undertaking, but we have been able to build and ship significant, consistent, integrated, and yet extensible, management capabilities covering a broad range of managed objects. The ability for DECmcc to adapt to the changing management environments underscores the benefit of adopting an architected approach to implementation.

8 Acknowledgments

The authors would like to acknowledge the work of the many people in the groups, past and present, responsible for bringing the ideas presented in this paper into practical reality in the DECmcc product set. Also, the detailed comments of two anonymous reviewers were very helpful.

9 References

1. M. Sylor, "Managing DECnet Phase V: The Entity Model," IEEE Networks (March 1988): 30-36.

2. Sylor, F. Dolan, and D. Shurtleff, "Network Management," Digital Technical Journal, vol. 5, no. 1 (Winter 1993, this issue)

3. EMA Entity Model (Maynard MA: Digital Equipment Corporation, Order No. AA-PV7KA-TE, January 1993).

4. DNA (Phase V) Common Management Information Protocol Functional Specification (Maynard MA: Digital Equipment Corporation, Order No. EK-DNA01-FS-001, July 1991).

5. DNA (Phase V) Network Control Language Functional Specification (Maynard MA: Digital Equipment Corporation, Order No. EK-DNA05-FS-001, July 1991).

6. L. Fehskens, "An Architectural Strategy for Enterprise Management," IFIP Proceedings of the First Symposium on Integrated Network Management (May 1989): 41-60.

7. M. Sylor, "Guidelines for Structuring Manageable Entities," IFIP Proceedings of the First Symposium on Integrated Network Management (May 1989): 169-183.

8. Information Technology: Open Systems Interconnection: Common Management Information Service Definition, ISO/IEC 9595 (Geneva: International Organization for Standardization/International Electrotechnical Commission, 1990).

9. Information Technology: Open Systems Interconnection: Common Management Information Protocol Specification, Part 1, ISO/IEC 9596-1 (Geneva: International Organization for Standardization/International Electrotechnical Commission, 1990).

10. N. La Pelle, M. Seger, and M. Sylor, "The Evolution of Network Management Products," Digital Technical Journal, vol. 1, no. 3 (September 1986): 117-128.

11. D. Shurtleff and C. Strutt, "Extensibility of an Enterprise Management Director," Network Management and Control, A. Kershenbaum, M. Malek, and M. Wall (eds.) (New York: Plenum Press, 1990): 129-141.

12. J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol (SNMP)," RFC 1157 (May 1990).

13.G. Stone, "Integrated Management Technologies," AT&T UNIX Systems Management Symposium, Spring 1991.

14.C. Strutt, "Dealing with Scale in an Enterprise Management Director," IFIP Proceedings of the Second Symposium on Integrated Network Management (April 1991): 577-593.

15. S. Martin, J. McCann, and D. Oran, "Development of the VAX Distributed Name Service," Digital Technical Journal, vol. 1, no. 9 (June 1989): 9-15.

16. A. Shvartsman, "An Historical Object Base in an Enterprise Management Director," IFIP Proceedings of the Third Symposium on Integrated Network Management (April 1993): 123-134.

17. C. Strutt and D. Shurtleff, "Architecture for an Integrated, Extensible Enterprise Management Director," IFIP Proceedings of the First Symposium on Integrated Network Management (May 1989): 61-72.

18. J. Borden, "Digital's Telecommunications Network Management Program," Network Operations and Management (New York: The Institute of Electrical and Electronics Engineers, 1992): 102-111.

19. DECmcc System Reference Manual, 2 volumes (Maynard MA: Digital Equipment Corporation, Order No. AA-PD5LC-TE, AA-PE55C-TE, April 1992).

20. J. Fernandez and K. Winkler, "Modeling SNA Networks using the Structure of Management Information," IEEE Communications (May 1993).

21. C. Strutt, "Distribution in an Enterprise Management Director," IFIP Proceedings of the Third Symposium on Integrated Network Management (April 1993): 223-234.

10 Biographies

Colin Strutt Colin Strutt is the DECmcc technical director in Enterprise Management Frameworks, part of the NAS Systems Management. Prior to that position, Colin was the project leader for the terminal server manager, various terminal server products, Ethernet communications server, and DECnet-IAS. He joined Digital in 1980 and is now a consulting engineer. Colin received a B.A. (honors) and a Ph.D. both in computer science from the University of Essex, UK. He is a member of BCS and ACM. Colin has several patents pending on DECmcc technology and has published papers on integrated network management.

James A. Swist Jim Swist joined Digital in 1975. He is a consulting software engineer and the technical leader for open systems in the Enterprise Management Frameworks Group. Prior to this position, he was a system management architect for VMS development, technical leader and development manager for TDMS/ACMS/CDD database systems, and a consultant in software services for several large commercial TP projects. Jim earned a B.S. in electrical engineering from the Massachusetts Institute of Technology in 1970. He has one patent pending on MCC distributed dispatch.

11 Trademarks

The following are trademarks of Digital Equipment Corporation:

DECmcc, DECnet, DECbridge, DECconcentrator, DECthreads, Digital, LANbridge, OpenVMS on Alpha AXP, OSF/1 on Alpha AXP, POLYCENTER, POLYCENTER Network Manager 200, POLYCENTER Network Manager 400, POLYCENTER SNA Manager, ULTRIX, VAX, and VMS.

Advanced System Management and SOLVE: Connect for EMA are trademarks of System Center, Inc.

IBM and NetView are registered trademarks of International Business Machines Corporation.

Motif and OSF are registered trademarks of Open Software Foundation, Inc.

System V is a trademark of American Telephone and Telegraph Company.

UNIX is a registered trademark of UNIX System Laboratories, Inc.